N88-17255

THE KATE SHELL- AN IMPLEMENTATION OF MODEL-BASED CONTROL, MONITOR AND DIAGNOSIS

July 1987

Matthew Cornell

Artificial Intelligence Office (mail stop: DL-DSD-22)
NASA/Kennedy Space Center, Florida 32899 (305) 867-3224

ABSTRACT

The conventional control and monitor software currently used by the Space Center for Space Shuttle processing has many limitations such as high maintenance costs, limited diagnostic capabilities and simulation support. These limitations have driven us to develop a knowledge-based (or model-based) shell to generically control and monitor electro-mechanical systems. The knowledge base describes the system's structure and function and is used by a software shell to do real-time constraints checking, low-level control of components, diagnosis of detected faults, sensor validation, automatic generation of schematic diagrams and automatic recovery from failures. This approach is more versatile and more powerful than the conventional "hard coded" approach and offers many advantages over it, although, for systems which require high speed reaction times or aren't well understood, knowledge-based control and monitor systems may not yet be appropriate.

INTRODUCTION

Background of Problem

The current Launch Processing System (LPS) at the Kennedy Space Center (KSC) has the duty of checking-out and launching Space Shuttle vehicles. The responsibility begins with testing the Orbiter's many systems in the Orbiter Processing Facility, continues through the assembly and check-out of the integrated vehicle (External Tank, Solid Rocket Boosters and Orbiter) and ends with final countdown and launch. The computing system which supports these tasks is a decade-old network of 15 consoles (1 console = 1 computer & 3 monitor/keyboard units), a huge data-communications system (Front End Processors and Common Data Buffers), and large Mainframes for data storage and retrieval, documentation and configuration control. The console computers themselves have a 16-bit address bus, 64K of RAM and a single 80Mb Winchester disk. To provide a feel for the capability of this system, the total number of "Function Designators" (FD's) in the KSC database, which describe all commands and measurements, is approximately 95,000 including Cargo, Facility, Ground and Flight FD's. The actual number used for each launch is in the neighborhood of 40,000.

LPS (Conventional) Software Approach

The software applications running on the console computers used to check-out the Space Shuttle vehicle are written in a NASA-designed procedural language called Ground Operations Aerospace Language or GOAL. These GOAL procedures are conventional control and monitor programs specific to the system the writer/engineer is responsible for and vary widely from system to system and even within a single system (due to the fact that many people participate in writing system test programs). The typical program is "hard-coded" to the components and FD's of the system it controls and monitors, specifying exactly what steps to take in order to command any component to any desired state.

LIMITATIONS OF LPS/CONVENTIONAL APPROACH

Limited Diagnostic Capability

The program's only ability to perform diagnostics are of the failure tree sort which maps a restricted number of possible failure conditions (pattern of measurements) to a probable cause, determined by the engineer and then coded in GOAL. Naturally, a system of reasonable size makes the ability to do a complete tree search impossible. The number of possibilities grows exponentially with each new component added.

Limited Display Capability

The user interface consists of character-based "display skeletons" with cursor keys for pointing and issuing commands. Systems are displayed as well as possible but the displays can not be dynamically reconfigured and are limited in number for each console. As in many applications today, much of the GOAL code is written to handle the user interface.

Limited Data Retrieval Capability

Data Retrieval is through a command-line interface with unforgiving syntax. Graphs of stored data are available but necessarily restricted due to the character-based display and screen size.

Low-to-Medium Fidelity Simulation

For application program testing and training, simulation is done off-line by software engineers on a system called Shuttle Ground Operations Simulator (SGOS) in a custom simulator language. One problem is that the software engineers are often not familiar with the true operation of the hardware and it takes many iterations to get even the more basic functions working correctly.

PROBLEMS RESULTING FROM CONVENTIONAL APPROACH

Expensive to Maintain

Because each application program is specific to its system, any modifications or corrections in the equipment require the system engineer to individually change all applications and SGOS programs and retest and reverify them (usually a long process) before they are operating correctly.

No Sensor Validation

One of the biggest and most-encountered problems seen during check-out of many systems is that of sensor validation. Many launches and tests have been delayed or aborted due to a sensor indicating a faulty component when, in actuality, the component was fine and the sensor or measurement channel itself was at fault. Some diagnostics can be written to help with this problem but, as stated earlier, it is impossible to handle all cases. Delays of this sort in the Space Program have equated to millions of dollars.

Additional LPS/Space Center-Related Problems

Two additional problems, experienced system engineers reaching retirement age and accelerated launch schedules, are leaving relatively inexperienced personnel to monitor and run the check-outs. These, combined with the problems above, result in both delays and errors in the day-to-day processing.

Obviously, out-dated equipment plays a role in some of the problems listed. But replacing the hardware with no change in software except for the ability to have larger programs will not solve these problems. What is required is a much more intelligent control and monitor approach. We believe the knowledge-based or model-based techniques described here and exhibited in the Kennedy Space Center's Knowledge-based Autonomous Test Engineer project (KATE) are a viable solution.

The remainder of this paper will cover the following topics:

The Model-based Approach- A discussion of our understanding of model-based control, monitor and diagnosis and how we've applied these techniques to some of the problems at KSC.

The KATE Implementation- A short status of the KATE project describing its capabilities.

Limitations- A description of the kinds of systems and situations the KATE shell may not be applicable to.

Future- A summary of some projects related to the KATE technology and how we hope KATE will affect these other programs.

THE MODEL-BASED APPROACH

Since our understanding of this approach is from an engineering perspective, not a fully comprehensive, academic one, this discussion will necessarily be limited to the work done in developing the KATE shell. Davis [1] and de Kleer [2] provide more complete information on model-based systems. The model-based approach is founded on the idea of describing a system of components in terms of its structure and function. Structure is how the components of the system are interconnected, for example "The output of transistor one is connected to the inputs of transistors two and three". Function, on the other hand, tells us how the component operates in terms of relating inputs to outputs, i.e. a transfer function, and tells us how the component can be expected to perform. For example "The output of transistor one is 2.5V when the input is above or equal to 1.0V and the output is 0.0V when the input is below 1.0V". This explicit description allows us to effectively separate the system-specific knowledge from the procedural "smarts" which reason about the system's operation from the knowledge base. This dichotomy means physical system changes, such as replacing existing components or adding new components (both of which might change the behavior of the system) require a very easy knowledge base change without impact to the procedural portion (the shell) of the system. This assumes the shell is robust enough to handle the new component.

This approach is markedly different from rule-based control and monitor systems which are very specific to a particular system. The underlying approach to control, monitor and diagnosis employed by these systems is really no different than the conventional approach described above. For diagnosis, rule-based systems use rules about the components to map a particular pattern of measurements and states into a probable cause. They don't really reason about why the answer is probable; such knowledge is simply written down by the engineer (where the real understanding is) and put into rules. Control is done in a similar manner. The options for getting a component to a desired state are decided by the engineer (using his internal model of the system and reasoning skills) and then committed to a rule-base. The major advantages offered by modern rule-based systems are their highly-developed user interfaces (they usually run on Lisp Machines) and development and graphics tools.

Having the system described in a KATE-style knowledge-based format allows our shell to do the same kinds of things engineers do with the information.

Monitoring

Since the shell knows the values of all commands which control the system and how the components of the system work, it can derive the expected values of all objects (including measurements) by "propagating" the effects from the commands down through the knowledge base to the measurements. If these expected measurement values don't match the measured values coming in from the system there is either a malfunctioning component or an inaccurate knowledge base. Hopefully, the system's knowledge base has been carefully developed and matches the real hardware.

Diagnosing

One of the most desirable operations that can be performed on a system is that of diagnosis, the process of determining which components are causing the system to behave in an incorrect manner. The way our shell's diagnoser does this is by gathering a list of all objects which could affect the discrepant measurement (see "Monitoring" above) and eliminating those whose failure can't account for the anomaly. (We liken this process to that of a "mind game"- "If object A failed then it would cause B, C and D to read b, c and d. Since they're not, A must be OK"). If, after processing all controlling components only one is left, then it must be the culprit. Otherwise you're left with a list of suspects, any one of which could be failing. (Our diagnoser only checks for single points of failure, not simultaneous, unrelated multiple failures. The claim can be made that, at least with most systems at KSC, single point failures are the norm and simultaneous, unrelated multiple failures rare [3]). For more information on diagnosing see Scarl [4].

Simulating and Training

Because the knowledge base describes the system's components and operation as accurately as possible, our system is by definition a high-fidelity model. Our shell uses this model inherently in monitoring and diagnosing but has another copy of the knowledge base which runs in parallel to the first and acts as the real-world system. Objects (components) can be failed and the simulator process will propagate the failure through its knowledge base, sending the affected measurements to the constraints (monitoring) system just like the real system would. This allows you to develop the knowledge base (and currently for us the shell) without being physically connected to the system. Also, operators can be trained on this system which (if modelled correctly) will perform just as if connected to the hardware.

Controlling

In traditional control and monitor software the methods of controlling each component of interest are directly and explicitly coded. With the approach used in KATE, a more declarative style of control is available. Instead of telling the program exactly how to accomplish the control goal you can simply request any object (command, measurement, component or pseudo-object such as an internal pressure) to be in any state. The control portion of the shell searches backwards in the knowledge base from the component to the controlling commands, building a context-sensitive (uses current component states) math expression along the way. This math model is used to construct any and all possible combinations of commands that would give you the desired state. Any combinations which violate previous requirements (see "Maintaining" below) are not allowed. Currently, the KATE shell automatically uses the option which requires the fewest number of commands*. What this means is the low-level (component-level) control is automatically done for you. Upon this low-level base will be built a very high-level control language whose statements would read just like a procedure describing the desired operation of the system. For example: "Maintain AFT-DUCT-FLOW at 25 LBS/MIN for 15 MINUTES".

Maintaining

If it is desired to maintain a particular object in a particular state (like a purge-pressure at some value) you can get it to that state and keep the fact that it's to be maintained on a list. In the future, any diagnosis that indicates the failed component(s) effect that maintained object, the control code can be automatically run again to reinstate that now-violated request. (Recall the control portion is context-sensitive which includes not using failed components).

Drawing

The structure information in the knowledge base gives you much of what you need to draw engineering-style schematic diagrams. Once drawn, the actual values for each measured component (as well as inferred values for all others) can be displayed dynamically, creating a "live" drawing. If system-grouping information is supplied or can be inferred, the displays can be on a functional level with the operator able to request the "level" of viewing. For example, a valve with all its supporting circuitry, commands and measurements can be displayed by itself on the screen, or a number of valves with just their significant measurements (perhaps position) shown. The drawing also allows the user to do control by pointing and clicking the computer's mouse on any component and inputting a desired value.

Single Point Failure Analyzing

Since the control portion of the KATE shell accumulates all possible combinations of commands to achieve a particular goal, its routines can be run on each component to produce a list of components which are single points of failure, i.e. they operate without redundancy. This single point failure analysis program could be run dynamically after each diagnosed failure to give a running account of system vulnerability.

*This is not the final way KATE will chose which option to use; there are usually natural priorities and requirements that would constrain the selection and would be integrated into the control code.

Knowledge Base Generation from CAD Files

Because this entire approach is based on the knowledge base, it is vital that it is as correct and as complete as possible. To aid the system developer in this, much of the knowledge base creation task can be accomplished automatically by a program which uses CAD-generated drawing information.

Automatic Documentation/Retrieval

Since any information can be stored with each object in the knowledge base, it provides a very convenient way to have component documentation (specification sheets, failure histories, vendor/supplier data) and digitized video very easily accessible on-line.

THE KATE IMPLEMENTATION

Many of the features listed above have been embodied in the KATE project at the Kennedy Space Center. KATE is an expansion of an earlier joint MITRE/KSC project called LES, or LOX (Liquid Oxygen) Expert System. LES is a real-time monitor and diagnosis expert system which was used to successfully monitor six Space Shuttle LOX loadings at the Space Center. Scarl [4] and Scarl [5] describe LES's capabilities and methods in detail.

The KATE shell is implemented using a modified version of frames and the Frame Representation Language [6] to represent and access the knowledge base. It runs on both AT-class microcomputers (for demonstrations) and Lisp machines, the latter being used for further development. The code is written entirely in Common Lisp, with extensions used for the user interface and multi-tasking. As of this writing, the KATE shell exhibits monitoring, goal-seeking control/maintenance and diagnosis of failures for objects without state or feedback. In the control and diagnostic portion of the shell, work is being done to handle components which utilize feedback and internal state in their operation and control loops. Work has just begun to update KATE's drawing system to do completely automatic schematic diagram generation and display.

A plotting system is functioning which is smart about how to plot all measurements related to a particular measurement or component. Work is also progressing on a program to do single point failure analysis and generation of KATE-compatible knowledge bases from standard CAD drawing files.

Current development of the KATE shell is being done in order to provide an operational control, monitor and diagnostic system for the Environmental Control System (ECS) in the Orbiter Maintenance and Refurbishment Facility (OMRF) which is currently under construction. This building will be used for storing Space Shuttle Orbiters and doing maintenance work on them and the ECS system will provide a conditioned flow of air to four areas of the orbiter for the purposes of ventilation, cooling, and controlling static electrical discharge. Because the OMRF ECS system consists of many kinds of components not currently handled by KATE, the development of its knowledge base is driving expansion of the shell's capabilities. This will be the first operational real-time control and monitor expert system at KSC and it has the potential of being included in other projects at the Space Center.

LIMITATIONS

Although the systems the KATE shell has been applied to have been mostly electromechanical in nature (pumps, valves, relays, discrete and analog commands and measurements) it can be applied to many others. However, there are some cases and conditions where this approach might not be appropriate. (Those related to speed or efficiency might be addressed by faster computers or more efficient code):

Poorly Understood Systems- Since the KATE shell relies entirely on a very accurate description of the system, the systems which it can handle must be well understood in terms of identifying each component, how the components work and exactly how they are connected, for example an electrical circuit. A biological system (like the human body), since its functioning is not well understood, would not be a good candidate for this type of representation and modeling. However, you can model abstract concepts and you can model on different levels of abstraction with KATE, assuming you understand how those concepts or "boxes" interact and function on the level being modelled.

High Speed Systems- The systems we have applied KATE to have had significant measurement changes occurring about once per second. The reaction times displayed have been on the order of a maximum one second delay from the time when a measurement changes to the time when the shell can react with a command issuance. Diagnosis times are around four seconds for our most complex example* and we expect the maximum to drop to around one second.

"Broad" Systems- Since KATE's searching is depth-sensitive, it is most efficient for systems which have a small number of components between commands and measurements. Systems such as LOX fit this description with mostly four or five components between commands and measurements (one branch) but many copies of those branches repeated throughout the system for redundancy.

Poorly Instrumented Systems- Because KATE, like an engineer, uses supporting measurements for detecting faults and eliminating faulty components during a diagnosis, it needs good system visibility. The more measurements available, the better the shell (or an engineer) can detect the presence of a misbehaving component and determine exactly which component it is.

FUTURE

KATE could be applied to, and could have a positive impact on, many projects at the Space Center as well as in other control and monitor applications. What follows is a description of some possible application areas as well as a few future KSC projects and how they could benefit from model-based technology:

Ground Data Management System (GDMS)- GDMS is a major project being developed at KSC to support interface verification, integration and test activities for the various elements of the Space Station, from prelaunch on, at the Space Center. It will process and test Space Station "work package" items, interfaces broken for shipping, interfaces between different launch package items, and interfaces between the launch package and the orbiter. GDMS consists of a large multi-channel network system, each channel capable of handling about ten times as many commands and measurements as the current LPS. Each channel can support up to 256 mini/Mainframe computers acting as application processors and up to 256 engineering workstations for graphics and display processing. Each processor will also have access to large archival and retrieval subsystems, database systems,

*The LOX replenish valve is a very redundant subsystem which has around 20 components and 12 commands and measurements associated with it.

documentation processing, and other networks via gateways and bridges. The KATE shell could be the application doing the test and check-out job on the application processors. Since the elements will probably be very different from one another and will have to be tested before integration, trying to do so with conventional software technology would require a tremendous effort. Even if this is possible, the effort would not be applicable to other elements. With the knowledge-based approach, because only the knowledge base would have to be developed by the experimenter or vendor (probably from a CAD system), application code would never have to be written.

Heavy Lift Launch Vehicle- A check-out and control system will need to be created for the future heavy lift launch vehicle(s) being planned by NASA and the Air Force. The types of test and check-out tasks that need to be performed for this will be very similar to those done by LPS (and GDMS) and therefore could benefit greatly by employing some of the ideas in the KATE shell.

Electric Power and Research Institute (EPRI)- Currently, a NASA technology transfer program is underway with EPRI to apply the KATE shell to problems in the power industry and nuclear power plants in particular. Early investigation and interviews with plant managers have identified many areas where KATE could offer substantial savings and safety features.

Other Applications- Other areas which could benefit from the approach taken in KATE include on-board aircraft, ship and spacecraft systems, the Space Station, satellites and the National Aerospace Plane.

Conclusion

We feel that a knowledge-based control and monitor approach, such as exemplified by KATE, may be the only solution to problems where the systems under test are well understood, can be described in a structure-function format and it is desirable to do automated control, monitor, diagnosis, drawing, sensor validation and redundancy management on them. For large systems requiring diagnostics, traditional tree-structured methods are impractical, for identifying all possible failure states can easily become an impossible task. Having the low-level control performed automatically by the shell provides the basis for a very high-level procedural language which would eliminate or greatly simplify application program writing as we know it today, saving development time and money and providing features impossible to achieve with current methods.

REFERENCES

1. Davis, Randall, "Diagnostic Reasoning Based on Structure and Behavior", Qualitative Reasoning about Physical Systems, First Ed., MIT Press, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1986, pp. 347-410.
2. de Kleer, Johan, "How Circuits Work", Qualitative Reasoning about Physical Systems, First Ed., MIT Press, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1986, pp. 205-280.
3. Jamieson, John, "Single Point Failures at KSC", unpublished, DL-DSD-22, Kennedy Space Center, Florida, 32899.
4. Scarl, Ethan, Jamieson, John, Delaune, Carl, "Sensor-based Diagnosis using Knowledge of Structure and Function", IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-16 No. 6, November, 1986.
5. Scarl, Ethan, Jamieson, John, Delaune, Carl, "A Fault Detection and Isolation Method Applied to Liquid Oxygen Loading for the Space Shuttle", Proceedings 9th International Joint Conference Artificial Intelligence (IJCAI-85), pp. 414-416.
6. Roberts, R.B., Goldstein, I.P., "The FRL Manual", Massachusetts Institute of Technology AI Laboratory, Memo 409, September, 1977.

ACRONYMS

ECS	Environmental Control System
EPRI	Electric Power and Research Institute
FD	Function Designator
GDMS	Ground Data Management System
GOAL	Ground Operations Aerospace Language
KATE	Knowledge-based Autonomous Test Engineer
KSC	Kennedy Space Center
LES	LOX Expert System
LOX	Liquid Oxygen
LPS	Launch Processing System
OMRF	Orbiter Maintenance and Refurbishment Facility
SGOS	Shuttle Ground Operations Simulator